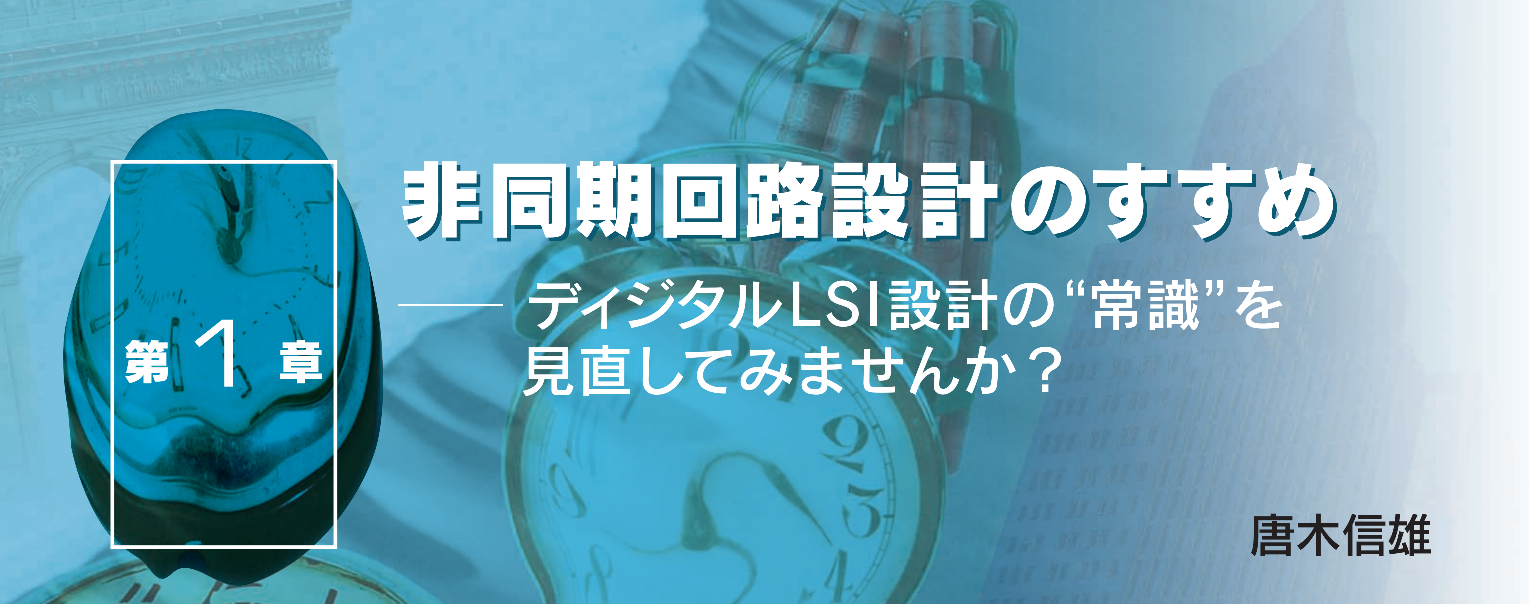


第1章

非同期回路設計のすすめ

――ディジタルLSI設計の“常識”を見直してみませんか？

唐木信雄

ここでは，非同期回路の設計技術について解説する．論理合成ツールが普及して以降，ディジタルLSI設計では同期設計方式が標準的なスタイルとなった．しかし，同期設計ゆえの不ぐあいも発生している．最近，もっとも大きな問題となっているのがクロック・スキューや電磁放射ノイズ，消費電力などの不ぐあいである．こうした問題を解消する手段の一つとして，クロックレスの非同期設計方式に注目が集まっている．ただし，現状で，非同期設計方式を前提に開発されたEDAツールは非常に少ない．開発環境や設計手法の整備などに大きな課題が残っている．（編集部）

最新の半導体製造プロセスを利用してディジタルLSIを開発する際の大きな課題として，クロック・スキューの増大，同時スイッチングなどに起因するノイズの増加，および消費電力の上昇などがあります．こうした問題を解消するため，ディジタルLSIの回路設計者やレイアウト設計者は大きな負担を強いられています．例えば，一つの不ぐあいの修正はチップ上のさまざまな箇所に影響を与えます．一度回路を修正するたびに作業の手戻りが発生し，なおかつ広範囲の回路を検証し直す必要があります．また，その修正作業が1回や2回で収束するとは限りません．

こうした問題が生じる根本的な原因は，現在のディジタルLSIが，一つまたは複数のクロック信号の立ち上がり/立ち下がりに合わせて回路全体が動作する「同期設計方式」に基づいて開発されているためです．同期設計方式は現在のディジタル回路設計の根幹です．論理合成技術の普及とともに，この方式はディジタル回路設計の標準的な手法として浸透しました．実際，最近になってディジタルLSI設計の世界に入ってきた技術者のみなさんの中には，同期設計以外の方式をよく知らない方もいるかもしれません（「非同期」イコール「ダメな設計」と教育されているケースもある）．

しかし，上述のように，半導体製造プロセスの微細化とともに，同期設計方式に起因する不ぐあいが生じるケースが目立ってきています．また，従来型の同期設計では解決できないタイプの問題（例えば，製造ばらつきのあるプロセスの利用を前提とした集積回路の開発など）も出てきています．

本稿では，筆者ら（セイコーエプソン）が注目している非同期設計方式の基本概念について説明していきます．論理合成中心の世界で育った最近のディジタルLSI設計者の方の中には「非同期」ということばに拒否反応を示される方もいるかもしれません．また，非同期設計があたりまえだった時代を憶えているベテラン設計者の方々には，「時代遅れの手法」と映るかもしれません．しかし，ノイズや消費電力のトラブルに対して，「問題のある箇所を探し，不ぐあいを起こしている箇所が検出されたらそのたびに修正する」という後手後手の対処法をいつまでも続けられる保証はないと思います．

このあたりでそろそろ，「同期設計方式」というディジタルLSI設計の“常識”を見直すことを検討してみてはいかがでしょうか？

● 同期・非同期はバケツ・リレーの方式の違い

まず，同期回路と非同期回路の違いについて説明します．

同期回路では，回路ブロックがクロック信号によって駆動されてデータを転送します．一方，非同期回路では，回路ブロックどうしがお互いに通信し合ってデータを送ります（**図1**）．

バケツ・リレーをイメージしてみてください（**図2**）．同期回路は，一定の時間間隔（クロック周期）で全員が同時に隣の人にバケツを手渡すイメージです．すべての人が同じ

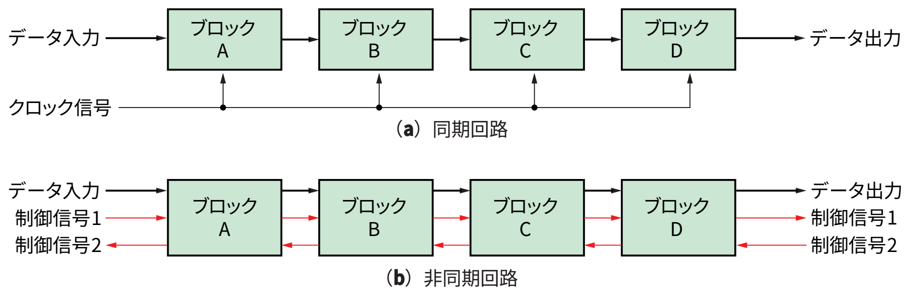


図1
非同期回路と同期回路におけるデータ転送
同期回路間のデータ転送はクロック信号によって駆動される．非同期回路間のデータ転送はブロック間の通信（制御信号）によって行われる．



(a) 同期式バケツ・リレー　(b) 非同期式バケツ・リレー

図2　バケツ・リレー
(a)の同期式バケツ・リレーでは，かけ声の速度が速くなると，みんなのタイミングについていけなくなる人が出てくるかもしれない．(b)の非同期式バケツ・リレーには，全体の統一感こそないが，もたもたした人がいても不ぐあいは起こらない．

タイミングで同じ動作を行います．手渡す速度が速く（クロック周波数が高く）なってくると，みんなのタイミングについていけなくなる人が出てくるかもしれません．

一方，非同期回路では，隣の人に「バケツをちょうだい！」（リクエスト），「受け取ったよ！」（アクノリッジ）と確認の声をかけながら，隣の人にバケツを手渡していきます．バケツを手渡す速さやタイミングは各自ばらばらです．もたもたしている人がいた場合でも，次の人は確認の声をきいてからバケツを受け取るので，トラブルは起こりません．全体の統一感はありませんし，毎回，確認が必要な点はめんどうなのですが，バケツ・リレーという作業そのものは滞りなく行われます．

どちらの方式にもそれぞれ一長一短があります．現在のディジタルLSI設計は，前者の方式においてバケツを手渡す速度が速すぎて，あちらこちらでトラブルが起こっている状態と考えることができます．

● なぜ今，非同期設計なのか

以下に，非同期設計方式の特徴を列挙します．

1) グローバル・クロックによって駆動されない

CMOS LSIの製造プロセスが0.25 μm以下になり，配線遅延（単位配線長当たりの配線容量に起因する信号遅延）が素子遅延（ゲート遅延）よりも大きくなりました．そのため，最近のディジタルLSIではクロック信号配線の総配線長が非常に長くなり，クロック・スキューの問題が深刻になってきています．また，LSIだけでなく，面積の大きい表示パネル（液晶ディスプレイなど）とともに形成される大面積の制御回路においても，同様の問題が発生しています．

一方，非同期回路には，そもそもクロック信号が存在しません．クロック・スキューの問題は発生のしようがありません．クロック動作に起因する電磁放射はなく，消費電力も下がります．例えば，筆者らが試作した非同期プロセッサでは，同期設計の場合と比べて消費電力が1/3，放射ノイズが－20dBになるという結果が出ています．

2) セルフタイムドな回路である

非同期回路は，各回路ブロックがそれぞれ固有の時間で動作するという意味で，“セルフタイムド（self-timed）”な回路と呼ばれています．処理時間の長い回路ブロックと短い回路ブロックが混在している場合でも，それぞれの回路ブロックはつねに最高速度で処理を実行します（ただし，チップ全体の性能が上がるかどうかは別問題．性能向上はチップ全体のアーキテクチャ設計のよしあしによる）．また，素子遅延や配線遅延にばらつきがあっても，問題なく動作します．これは動作電圧や温度などの変動に強いとも言え

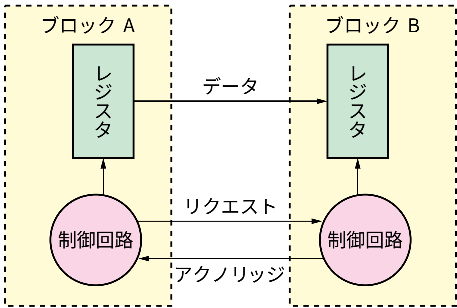


図3　ブロック間のハンドシェーキング

ブロックAが主導のAからBへのデータ転送を示している．まず，ブロックAはデータをデータ信号線に載せた後，リクエスト信号を出力する．次にBは，リクエスト信号を受け取った後，準備ができたところでデータ信号線上のデータを取り込み，アクノリッジ信号を出して応じる．Aは，アクノリッジ信号を受けてリクエスト信号を引っ込める．Bは，アクノリッジ信号を引っ込めることで応じる．この例ではAとBの間に3種類の信号が存在するが，データ信号線をリクエスト信号やアクノリッジ信号と兼用することで制御信号線を減らした2線符号化方式のハンドシェーキングもある．

ます注．

3）回路ブロック間はハンドシェーキングで通信

非同期設計方式の回路は，ハンドシェーキングによって通信し合う回路ブロックから構成されます（図3）．通信にはリクエスト信号とアクノリッジ信号を用います．回路ブロックのサイズを小さくすることで，処理の並列度を高めることができます．クロック間の周波数差や位相差の問題がないため，異なる製造プロセスへも比較的容易に移植できます．

4）イベントによって駆動される

非同期回路の回路ブロックはイベントによって駆動されます．イベントで駆動されるまで，回路ブロックは動作を停止しています．つまり，その間，電力を消費することはありません．また，起動時の待ち時間（レイテンシ）が少なくなります．同期回路ではワースト・ケースでタイミング設計を行う必要がありますが，非同期回路でその必要はありません．処理に要する時間（遅延時間）の平均値を用いてタイミング設計を行います．

5）制約を与える遅延モデルと開発環境が必要

非同期回路にはグローバル・クロックがありません．回路を意図したとおりに動作させるには，設計者自身がすべての回路のタイミングを考える必要があります．これはかなりたいへんな作業です．そこで，タイミングを取り扱いやすくするために，配線遅延や素子遅延に対してなんらかの制約を与える遅延モデルを規定します．こうした遅延モデルはいくつか提案されています．また，こうした遅延モデルに基づく設計・検証ツールも用意する必要があります．

表1　同期回路と非同期回路の比較

	同期回路	非同期回路
クロック関連	×	○
消費電力	×	○
電磁放射	×	○
プロセス移植	×	○
チップ面積	○	△または×
電圧・温度変動	×	○
素子特性ばらつき	×	○
設計環境	○	×

○：有利，△：やや不利，×：不利

同期回路と非同期回路の大ざっぱな比較を表1にまとめます．非同期回路の面積は，同期回路の場合と比べて2割～5割程度大きくなるという欠点があります．また，残念ながら，市販の設計・検証ツールのサポートがほとんどありません．

長年，多くの研究機関や大学などが非同期設計方式に関する研究を進めてきましたが，これまで非同期設計方式が商用LSIに全面的に採用されることはほとんどありませんでした．その理由の一つは，市販の設計・検証ツールが非同期設計方式をサポートしていないためだと思われます．実際のLSIの開発に非同期設計方式を導入する場合には，開発環境をどうするか（自前で構築するのか，あるいは数少ないながら登場し始めている非同期設計用の市販ツールを採用するのかなど）をきちんと検討することが重要になると思います．

● 非同期回路の全体像をつかむ

次に，非同期回路の分類を示します．同期回路でないものはすべて非同期回路と言えるのですが，この定義ではあまりにも漠然としています．また，人によって思い浮かべる構成や信号のふるまいが異なってしまうため，適切な議論ができません．そこで，いくつかの観点から非同期回路が分類されています．この分類をもとに，非同期回路の全体像をつかんでいただければと思います．

注：神経繊維やMEMSなどのメカ部品を構成要素とする回路系をイメージしていただきたい．高速な電気・電子素子や配線の遅延精度に比べて，神経繊維やメカ部品の応答速度はけた違いに遅く，かつ大きなばらつきが存在すると考えられる．同期設計の場合は，神経繊維やメカ部品を制御対象と見なして，いったん制御系から排除した回路構成を考えることになる．そのため，柔軟性やエネルギー効率の面で不利と言える．非同期設計であれば，このような回路系にもそのまま対応できる．

1）プロトコルによる分類

非同期回路では，回路ブロック間の通信にハンドシェーキング・プロトコルが使われます．これには，「4相ハンドシェーキング」と「2相ハンドシェーキング」があります．4相ハンドシェーキング・プロトコルはレベル（電位）ベース，あるいはRTZ（return to zero）ハンドシェーキング・プロトコルとも呼ばれます．一方，2相ハンドシェーキング・プロトコルはトランジション（遷移）ベース，またはNRTZ（non return to zero）ハンドシェーキング・プロトコルとも呼ばれます．それぞれの動作を**図4**に示します．

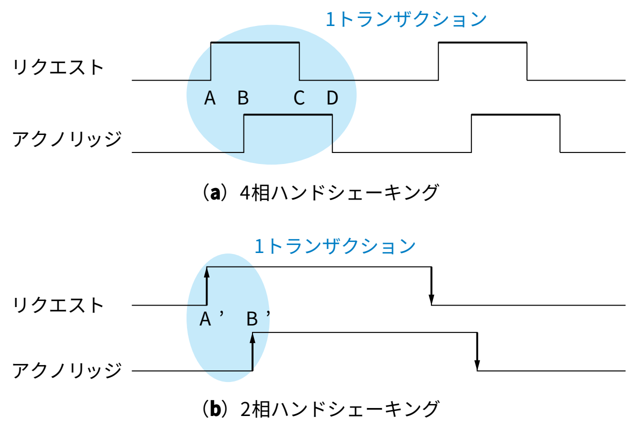


図4　ハンドシェーキング・プロトコルによる分類
（a）の4相ハンドシェーキングでは，まず主導側がリクエスト信号を出してシーケンスをスタートする（図の中のA）．これに対して受動側はアクノリッジ信号を出力して応じる（B）．主導側はリクエスト信号を引っ込めて応じる（C）．受動側はアクノリッジ信号を引っ込めて応じる（D）．（b）の2相ハンドシェーキングでは，まず主導側がリクエスト信号を変化させて，相手からの応答を待つ（A ’）．次に受動側がアクノリッジ信号を変化させてこれに応じる（B ’）．

2相ハンドシェーキングでは，かならず相手からの信号の変化を察知し，記憶しておくメカニズムが必要になります．しかし，同一の信号周波数の場合，4相ハンドシェーキングの2倍のトランザクションを受け渡せる可能性があります．

2）遅延モデルによる分類

非同期回路の遅延モデルとして，さまざまな手法が提案されています．例えば，「BD（bounded delay）モデル」，「DI（delay insensitive）モデル」，「SI（speed independent）モデル」，「QDI（quasi delay insensitive）モデル」などです．

BDモデルでは，すべての配線および論理素子に対して一定の範囲の遅延値（電圧や温度の変化に対応した最小値と最大値が規定される）があると仮定します．面積と消費電力をもっとも小さくできますが，安定的に動作させるためにはマージンを大きめにとる必要があります．

DIモデルでは，すべての配線および論理素子に対して任意の遅延（ゼロでも無限大でも，いかなる値でもかまわない）があると仮定します．スイッチングしさえすればいくら速くても遅くてもきちんと動作するような回路を設計する際に用います．どのような温度・電圧条件でも安定動作します．面積はもっとも大きく，消費電力もBDモデルと比べると大きくなります．

SIモデルは，すべての分岐配線について，配線の分岐先への伝播遅延時間はすべて等しい（全分岐端における信号遅延の差がほぼゼロ）と仮定したモデルです．レイアウトの際に分岐配線の配線長をそろえるなどの制約が生じます．

QDIモデルは，一部の分岐配線について，配線の分岐先への伝播遅延時間が等しいと仮定したモデルです．SIモデルよりはレイアウト時の制約が軽減されます．

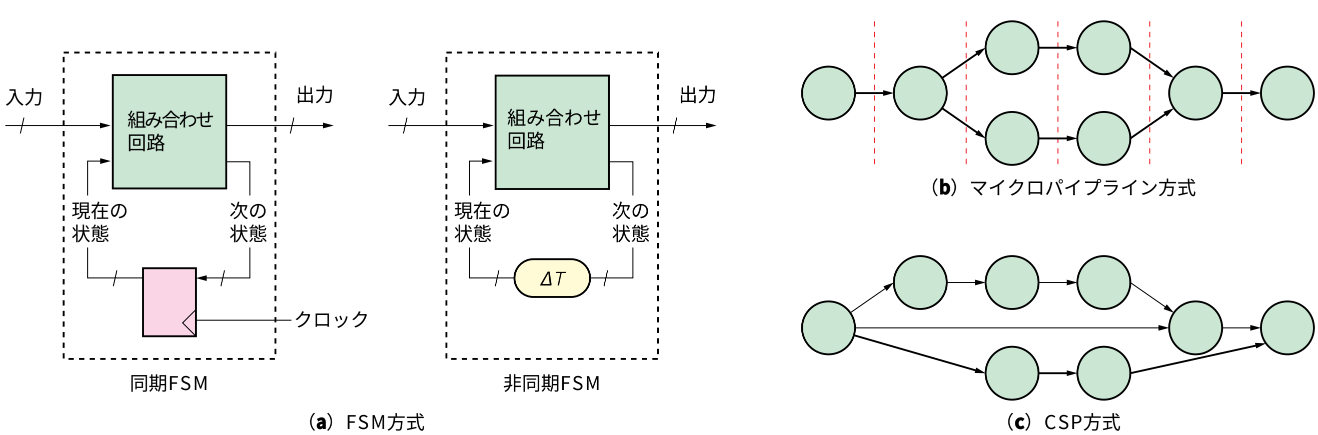


図5　構造による分類
非同期回路を取り扱いやすくするために，いくつかの回路構造のモデルが提案されている．そのうちの三つの例を示す．

3）構造による分類

非同期回路を取り扱う場合，回路構造の表現についてのいくつかの考えかたが提案されています．ここでは，FSM (finite state machine)方式，マイクロパイプライン方式，CSP(communication sequential process)方式について紹介します(図5)．

FSM方式では，一つまたは複数の非同期ステート・マシンによって非同期回路を構成します．同期回路の場合のステート・マシンと似た構造ですが，クロック信号がないので，状態遷移のタイミングを作るための遅延素子を挿入する必要があります．データパス回路はステート・マシンで作られる信号によって制御されます(これをバンドル・データ方式と呼ぶ)．ステート・マシンがつねに忙しく動いているため，比較的大きな電力を消費します．

マイクロパイプライン方式では，データがマイクロパイプラインの段(ステージ)から段へと順次送られていきます．グローバル・クロックがローカルな制御信号に置き換わっている点を除けば，同期設計のパイプラインとよく似ています．この方式のデメリットは，データがその処理を行わない段も通過しなければならない点です．

CSP方式は，互いに通信し合ってデータ転送を行うプロセスによって非同期回路を構成します．データは実際に必要とするプロセスにのみ転送されます．それ以外のプロセスはアイドル(待機)状態となり，電力を消費しません．

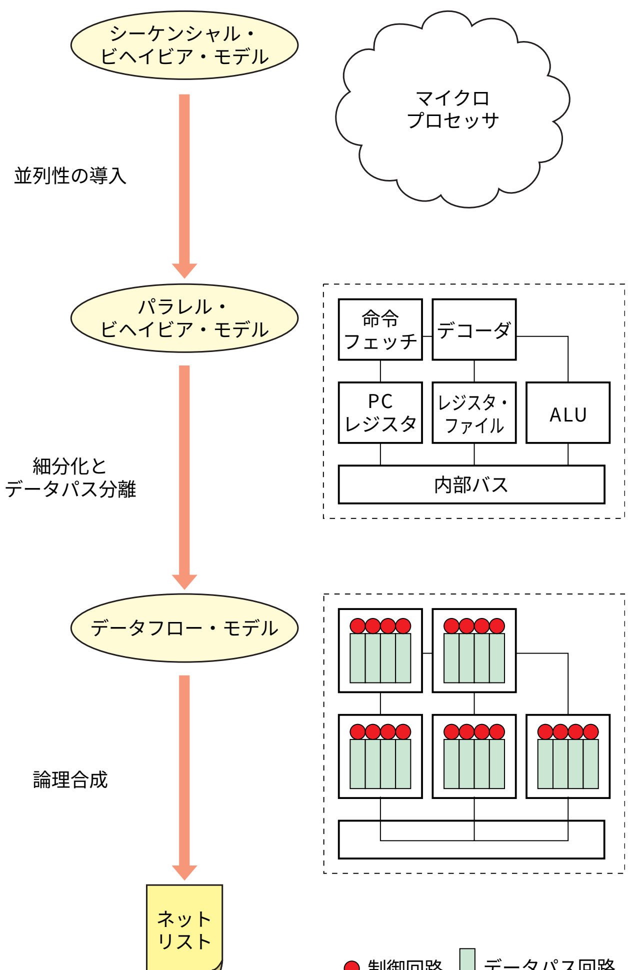


図6　非同期回路の設計フローの例

ここでは筆者らが採用しているCSP方式，4相ハンドシェーキングを想定したマイクロコントローラの設計フローを示している．チップ面積を抑えるため，遅延モデルとしてデータパス回路にはBDモデルを，制御回路にはQDIモデルを想定している．まず，非同期プロセッサのプログラミング・モデルに基づいて，Verilog HDLでシーケンシャル・ビヘイビア・モデル(アルゴリズム的に表現したビヘイビア・モデル)を記述し，機能検証を行う．次に，アーキテクチャに基づいて処理の並列性を導入する．筆者らが独自に開発した非同期回路記述言語(Verilog＋)でパラレル・ビヘイビア・モデル(並列処理部分を明示したビヘイビア・モデル)を記述する．トランスレータ(xlator)を用いてVerilog HDLに変換し，再度，機能検証を行う．その後，機能ブロックごとに回路を細分化して並列度を高める．同時にデータパスの分離を行い，データフロー・モデルを生成する．最後に，論理合成ツール(ctrlsyn)を用いて制御回路の論理合成を行う．データパス回路については，回路図入力で作成し，ネットリストを生成した．制御回路とデータパス回路のネットリストをマージし，ゲート・レベルの検証(論理検証)，タイミング検証，およびトランジスタ・レベルの消費電力解析を行う．

● 非同期回路についての素朴な疑問

ここで，非同期設計方式について筆者らがよく質問される項目を紹介します．

1）非同期回路の性能はどう示す？

パソコンの性能指標の一つとして，少し前まではマイクロプロセッサ(CPU)のクロック周波数が使われていました．技術的に見るとかならずしも厳密な指標ではないのですが，とてもわかりやすい指標と言えます．

このように同期回路では，回路構成が同じであれば，クロック周波数が高いほど処理性能が上がります．一方，非同期回路にはクロックがありません．非同期回路の性能はどのように表現するのでしょうか？

非同期回路の場合，ベンチマーク・テストを実施して，単位時間当たりの処理量を示すしかありません．CPUであれば，単位時間当りの実行命令数，すなわちMIPS(million instructions per second)などを使えばよいと思います．

2）回路を合成できるの？

研究機関や大学などでは，例えばCSPやTangram，Occamといった非同期回路の記述が可能な言語が提案されており，これに基づいた論理合成ツールが開発されています[1]．研究レベルで言えば，非同期回路の論理合成は行えると言えます．

しかし，市販の論理合成ツールを使って，設計者が意図

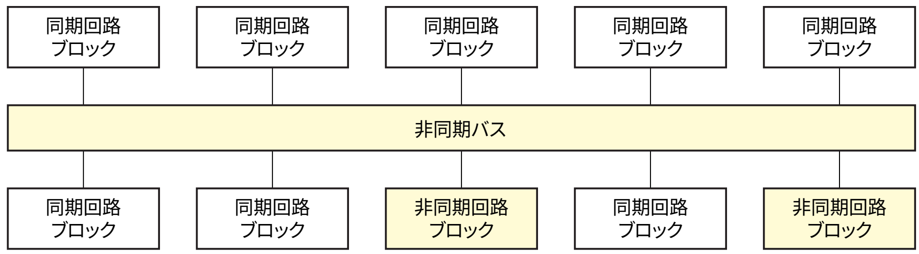


図7
GALS (globally-asynchronous locally-synchronous)
それぞれ固有のクロック信号によって駆動されるいくつかの同期回路ブロックが，非同期回路(非同期バス)によって接続されている．別々に開発された多数のIPコア(大規模回路ブロック)を搭載するシステムLSIなどに有効なアーキテクチャである．

したとおりの非同期回路を合成することは，不可能とは言いませんが，非常に困難です．非同期回路設計では，回路ブロック間の非同期通信機構を定義しなければなりません．つまり，非同期設計専用のプリミティブやマクロ・セルを用意し，論理合成ツールのライブラリに登録する必要があります．また，設計者はこのプリミティブを利用する箇所を逐一指示してやらなければなりません．

筆者らの場合，Verilog HDLのサブセットに非同期回路ブロック間の通信用プリミティブを追加した専用の回路記述言語(Verilog＋)やその言語に対応した論理合成ツールなどを開発しました．論理合成ツールは，ターゲットとなるライブラリと合成プログラムから構成されています．通常の論理合成ツールがサポートしていない`fork-join`ブロック，および通信コマンドまで含めた論理合成を行えます．

遅延モデルとして，データパス回路にBDモデル，制御回路にQDIモデルを採用した場合の非同期回路の設計フローの例を**図6**に示します．このフローでは，回路図入力ではなく言語を用いています．検証から論理合成まですべてテキスト・ベースで行う点は，最近の同期設計方式と同じです．

3) チップ全体を非同期にする必要があるの？

チップ全体を非同期にするかどうかは，設計者の考えかたによります．チップ全体を非同期にすることも可能ですし，一部だけを非同期にすることもありえます．例えば，同期回路と非同期回路が共存するGALS(globally-asynchronous locally-synchronous；大域的には非同期，局所的には同期)という回路構成があります(**図7**)．

GALS構成の回路では，それぞれ固有のクロック信号によって駆動されるいくつかの同期回路ブロックが，非同期回路(非同期バス)によって接続されます．動作周波数の異なる回路ブロックを搭載するシステムLSIなどがこのような構成を採ります．GALS構成では，同期回路ブロックを順次非同期回路ブロックに置き換えていき，最終的にすべてを非同期回路によって構成するということも可能です．

参考・引用*文献

(1) K. V. Berkel；*Handshake Circuits - An asynchronous architecture for VLSI programming*，Cambridge University Press，1993.

からき・のぶお
セイコーエプソン(株)　テクノロジー・プラットフォーム研究所

<筆者プロフィール>
唐木信雄．1972年に早稲田大学理工学部 電気工学科を卒業し，三井造船の会話型CADの開発に参画した．1980年にセイコーエプソンに入社し，漢字プリンタの設計に従事した後，インクジェット・プリンタ用カラー・モジュール開発などの責任者を務めた．1995年から5年間，米国Epson Research & Development社に赴任し，非同期回路設計の研究の指揮を執った後，現職にて研究に従事している．

1